IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

ORACLE CORPORATION and ORACLE AMERICA, INC., Plaintiffs, v. ADVANCED DYNAMIC INTERFACES, LLC, Defendant.	Civil Case No. 1:12-cv-01154-GMS JURY TRIAL DEMANDED
ADVANCED DYNAMIC INTERFACES, LLC, Plaintiff, v. SALESFORCE.COM, INC., Defendant.	Civil Case No. 1:13-cv-00397-GMS JURY TRIAL DEMANDED

OPENING CLAIM CONSTRUCTION BRIEF OF ORACLE CORPORATION, ORACLE AMERICA, INC. AND SALESFORCE.COM, INC.

Mary B. Graham (#2256)
Stephen J. Kraftschik (#5623)
MORRIS, NICHOLS, ARSHT & TUNNELL LLP
1201 N. Market Street
P.O. Box 1347
Wilmington, DE 19899-1347
(302) 658-9200
mgraham@mnat.com
skraftschik@mnat.com

Attorneys for Oracle Corporation and Oracle America, Inc.

Jack B. Blumenfeld (#1014)
Jennifer Ying (#5550)
MORRIS, NICHOLS, ARSHT & TUNNELL LLP
1201 North Market Street
P.O. Box 1347
Wilmington, DE 19899
(302) 658-9200
jblumenfeld@mnat.com
jying@mnat.com

Attorneys for salesforce.com, inc.

April 8, 2014

TABLE OF CONTENTS

Page

TABLE OF AUTHORITIES

Page(s)

CASES

I. NATURE AND STAGE OF THE PROCEEDINGS

Oracle Corporation and Oracle America, Inc. (collectively, "Oracle") filed a declaratory judgment action against Advanced Dynamic Interfaces, LLC ("ADI") regarding the patents-in-suit[1] on September 18, 2012. *See* C.A. No. 12-1154, D.I. 1. On March 11, 2013, ADI filed an action against salesforce.com, inc. ("Salesforce") alleging infringement of the patents-in-suit. *See* C.A. No. 13-397, D.I. 1. The Oracle and Salesforce actions have been coordinated for *Markman* purposes. Oracle and Salesforce respectfully submit their opening claim construction brief addressing the disputed terms requiring construction by the Court.[2]

II. SUMMARY OF THE ARGUMENT

ADI proposes claim constructions that run afoul of controlling Federal Circuit precedent, all in an effort to stretch the patents-in-suit to cover products and methods that were never disclosed or claimed. ADI's constructions impermissibly divorce the disputed terms from their context within the claims, improperly rely on general purpose dictionary definitions that are inconsistent with the intrinsic evidence, and ignore the specification's limiting description of "the present invention." *See Verizon Service Corp. v. Vonage Holdings Corp*., 503 F.3d 1295, 1308 (Fed. Cir. 2007) (holding that when a patent "describes the features of the 'present invention' as a whole, this description limits the scope of the invention"). By contrast, Oracle and Salesforce's constructions should be adopted because they accurately reflect the scope of the claimed inventions, viewed in light of the disclosure in the specification and the context-relevant extrinsic evidence. "'The construction that stays true to the claim language and most naturally aligns with the patent's description of the invention will be, in the end, the correct construction.'"

[1] The patents-in-suit are U.S. Patent No. 7,062,502 ("the '502 patent") and U.S. Patent No. 7,401,094 ("the '094 patent").

[2] Oracle and Salesforce contend that the term "appropriate," as used in both patents-in-suit, is indefinite, but do not request that the Court address construction of that term at this time. *See infra* at section V.D.

Phillips v. AWH Corp., 415 F.3d 1303, 1316 (Fed. Cir. 2005) (*en banc*) (quoting *Renishaw PLC v. Marposs Societa' per Azioni*, 158 F.3d 1243, 1250 (Fed. Cir. 1998)).

III. LEGAL PRINCIPLES

The Federal Circuit has instructed that when construing a claim, significant weight should be afforded to the intrinsic evidence (the claims, the specification, and the prosecution history). *Vitronics Corp. v. Conceptronic, Inc.*, 90 F.3d 1576, 1582 (Fed. Cir. 1996) ("[I]ntrinsic evidence is the most significant source of the legally operative meaning of disputed claim language."). Such evidence constitutes the "public record of the patentee's claim, a record on which the public is entitled to rely." *Id.* at 1583.

In considering the intrinsic record, the claims themselves "are 'of primary importance, in the effort to ascertain precisely what it is that is patented.'" *Phillips*, 415 F.3d at 1312 (quoting *Merrill v. Yeomans*, 94 U.S. 568, 570 (1876); *id.* at 1314 ("[T]he context in which a term is used in the asserted claim can be highly instructive."). For instance, a patentee's decision to use different claim terms, particularly within a single clause of the claims, creates a presumption that those terms have different meanings. *Helmsderfer v. Bobrick Washroom Equip., Inc.*, 527 F.3d 1379, 1382 (Fed. Cir. 2008).

Claim terms also must be interpreted in the broader context of the teaching of the specification. *Phillips*, 415 F.3d at 1315 ("The specification is always highly relevant to the claim construction analysis. Usually, it is dispositive; it is the single best guide to the meaning of a disputed term.") (internal quotation marks omitted). In this endeavor, understanding the problem that the inventors were seeking to solve and their proposed solution is often highly probative. *See Voice Techs. Group, Inc. v. VMC Systems, Inc.*, 164 F.3d 605, 615 (Fed. Cir. 1999) (finding the inventor's "explanation of the problems that existed at the time the invention was made and the inventor's solution to these problems" relevant to the claim

construction analysis). In addition, the specification may use language which signals that a specific meaning is intended. For example, when a patent uses the phrase "the present invention," and then proceeds to describe components and characteristics of "the present invention," "this description limits the scope of the invention." *Verizon*, 503 F.3d at 1308.

Extrinsic evidence, especially in the form of technical dictionaries, may also be considered and "can shed useful light on the relevant art." *Phillips*, 415 F.3d at 1317-18. However, any extrinsic evidence must be tied to the context of the invention disclosed and claimed in the patents-in-suit and cannot change the meaning of a term as established by the intrinsic record. *Phillips*, 415 F.3d at 1322 ("[A] general-usage dictionary cannot overcome art-specific evidence of the meaning of a claim term."); *id.* at 1321 ("[H]eavy reliance on the dictionary divorced from the intrinsic evidence risks transforming the meaning of the claim term to the artisan into the meaning of the term in the abstract, out of its particular context, which is the specification.").

IV. STATEMENT OF THE FACTS

The patents-in-suit describe a specific way, through software, to automate certain aspects of creating a custom graphical user interface for a relational database. *See* '502 patent at Abstract, 1:15-20.[3] A "graphical user interface" is, and has been for decades, the common method by which a person interacts with a computer using a mouse or a touch interface, e.g., through visual menus, drop-down boxes, and other graphical mechanisms. It stands in contrast to a textual interface, in which all commands are entered solely by typing them into a console. A "relational database" is a specific type of database that stores data in tables that are "related" to each other. '502 patent at 17:30-33 (describing "various popular relational database systems,

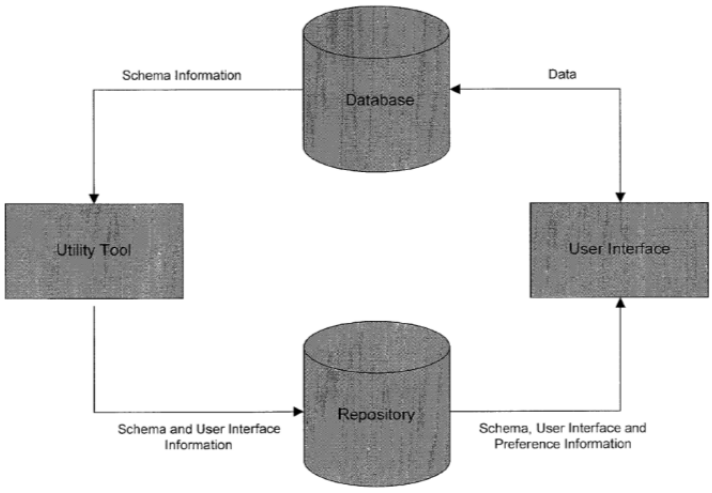
[3] The '094 patent issued from a continuation of the application for the '502 patent, and thus the patents-in-suit share a similar specification. For purposes of simplicity, all citations are made to the '502 patent, except where otherwise noted.

such as Microsoft SQL Server and Oracle"); Table 1 (defining "relational database"). For example, a "customer" table for a sales database might contain columns (alternately known as "fields") for customers' names, addresses, telephone numbers, etc., and each entry in a row of data (collectively known as a "record") contains that information about a particular customer. *Id.* at 1:27-28; Table 1 (defining "field" and "record"). A separate "customer order" table might include information about each order made, including, for example, the date ordered and the date shipped. The two tables might be related to each other by the inclusion of an entry in each record in the customer order table that identifies the ordering customer and refers back to the customer table. *See* Fig. 4 (showing the relationship between the top-left tables "CUSTOMER" and "CUSTOMER_ORDER," linked by the "CUSTOMER_ID" field); Fig. 84 (illustrating the addition to the "CUSTOMER_ORDER" table of a new "REQUESTED_DATE" field). In this way, the information about the customer (name, address, etc.) can be tied to one or more orders without storing the customer information repeatedly for every order the customer makes.

As a whole, the collection of tables and their relationships to each other are known as the "schema" of the database. *Id.* at Table 1 (defining "schema")[4]. Information about the schema is referred to in the patents as "metadata"—i.e., it is data generated by the claimed software about the structure of the database, as opposed to the actual data in the rows of the tables. *Id.* at Table 1 (defining "metadata"). The patents also use the term "metadata" to refer to information that the claimed user-interface generation software both produces and then uses to generate a custom user interface, such as data-entry forms and menus. *Id.* at 8:6-11. For example, the claimed software might generate specific user-interface metadata which it then uses to produce a user interface that displays data in a particular order. *See id.* at 22:11-18.

[4] More particularly, the "schema" is a "collection of database objects that includes tables, views, indexes, and foreign keys." *Id.* at Table 1.

At a high level, the patents are directed to generating metadata representing the structure of a relational database ("schema"), storing the metadata in a distinct repository, and using this metadata to generate a user interface. The patents set forth this architecture (referred to in the specification as "the present invention") in Figure 2:



Id. at Fig. 2.[5] The diagram of the invention depicts (1) extracting information about the structure of the relational database ("Schema Information" at upper left), (2) generating metadata about the schema and about the graphical user interface ("Schema and User Interface Information," at bottom left) and storing them in a repository (at bottom) distinct from the database, and (3) creating a graphical user interface "automatically" from the stored metadata. *Id.* at 8:1-20.

The patents claim that "numerous advantages" flow from generating schema metadata and user interface metadata and storing these metadata in the separate configuration repository. *Id.* at 8:1-20. These advantages include the ability to generate multiple user interfaces with a single instance of the claimed software, *id.* at 39:52-55 and Fig. 76, and the flexibility to develop interfaces for relational databases from different database software vendors, *id*. at 17:26-41.

[5] To provide a more legible image, the figure presented above has been captured from the file history of '502 patent, produced at ADI000269, rather than from the issued '502 patent. The content of the figures is identical.

V. ARGUMENT

A. "repository" ('502 patent, independent claims 1, 7, and 13; '094 patent, independent claims 1, 8, and 15)

Oracle and Salesforce's Proposed Construction	ADI's Proposed Construction
"memory structure distinct from the relational database for storing metadata"	"a place where data is stored"

All the independent claims of the patents-in-suit require a "repository" for storing schema metadata and graphical user interface metadata. The parties dispute whether the repository must be distinct from the relational database for which the graphical user interface is to be generated (Oracle and Salesforce's construction), or whether the repository could be *any* structure where data could be stored, including that same relational database mentioned elsewhere in the claim (ADI's construction).

The plain and ordinary meaning of the claim language requires that the metadata be stored in a repository that is separate and distinct from the relational database. Claim 1 of the '502 patent describes two distinct structures, namely, a "relational database" and a "repository":

> utility software ***extracting schema information from the relational database*** and automatically generating corresponding schema and graphical user interface ***metadata stored in a repository***; …

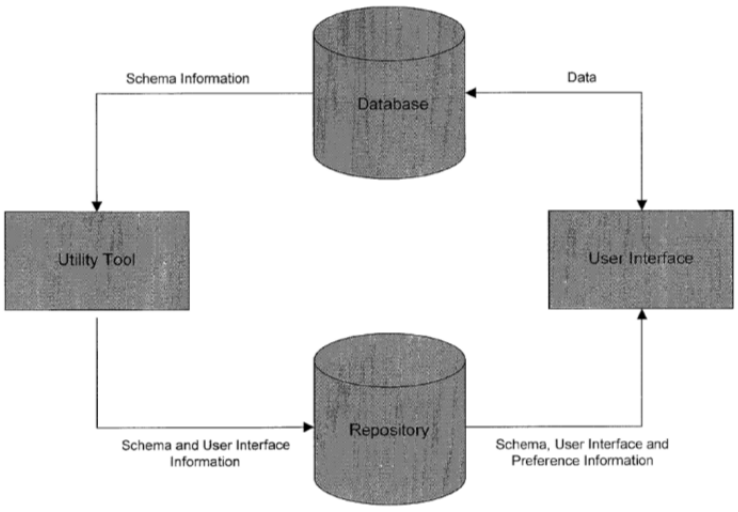
'502 patent, claim 1. The patentee's use of the terms "relational database" and "repository" in the same claim, particularly in the same clause, suggests that the terms refer to different structures. *See Ethicon Endo–Surgery v. U.S. Surgical Corp.*, 93 F.3d 1572, 1579 (Fed. Cir. 1996) ("If the terms 'pusher assembly' and 'pusher bar' described a single element, one would expect the claim to consistently refer to this element as either a 'pusher bar' or a 'pusher assembly,' but not both, especially not within the same clause.") (italics omitted); *Helmsderfer,* 527 F.3d at 1382 ("[D]ifferent claim terms are presumed to have different meanings."). Had the patentee intended that "repository" cover the "relational database" and any other storage device,

as ADI's construction would permit, the patentee easily could have drafted the claim language to make that clear, as illustrated below:

> utility software extracting schema information from the relational database and automatically generating ***and storing*** corresponding schema and graphical user interface metadata ~~stored in a repository~~;

As shown above, by inserting "and storing," and by deleting "stored in a repository," the claim would have made clear that the storage location was not important and could have included the database from which the schema information was extracted. But the patentee did not do so. Instead, the claims as written require that the metadata be stored "in a repository." *Id.* at claim 1.

The specification uniformly supports Oracle and Salesforce's interpretation that the "relational database" and the "repository" are distinct. The '502 patent explains that "the ***present invention*** proposes the high level architecture depicted in Fig. 2," which clearly illustrates the separation of the repository and the relational database. *Id.* at 8:1-2, Fig. 2.



The Federal Circuit has explained that "[w]hen a patent thus describes the features of the '***present invention***' as a whole, this description limits the scope of the invention." *Verizon*, 503 at 1308 (emphasis added). The patents explain that "under this architecture" depicted in Figure 2—which is the sole embodiment disclosed in the specification—information regarding database

structure (the "schema") is obtained first from the relational database, and then corresponding schema metadata and graphical user interface metadata are stored in the configuration repository. *Id.* at 8:2-8 ("Information gleaned from the schema is used to populate the configuration repository with metadata."); Fig. 2.

ADI's proposal that "repository" should be construed as any "place where data is stored" improperly broadens the claims. Contrary to this construction, Figure 2 does not show metadata being stored in the relational database itself, and the patents never describe storing schema and graphical user interface metadata in that same database. In fact, the patents explain the opposite: that storing schema and user interface information in a distinct configuration repository, and ***not*** in the relational database from which it was extracted, is central to the patents-in-suit:

> A ***key principal behind the present invention*** is that the structure, or schema, of a relational database provides a significant amount of useful information with respect to the data entry user interface. ***While it is theoretically possible to generate a user interface directly from the database schema without the use of a configuration repository, the layer of abstraction offered by the repository has numerous advantages.*** These advantages will be explored in depth in the remainder of this document.

Id. at 8:12-20.

The specifications then describe two advantages of a distinct configuration repository. First, by saving metadata in a distinct configuration repository and not in the relational database, a single instance of the user-interface generation software can be used to develop multiple interfaces. *See id.* at 39:52-55, Figs. 75-76 (listing different configuration repository files for "Customer Orders" and "Employee Expense Reports" that are available for use by the user-interface generation software). Second, by having a repository separate from the relational database, it is simpler to adapt the claimed invention to develop interfaces for different vendors' databases (e.g., Oracle's and Microsoft's). *Id.* at 2:34-39, 12:67-13:2. ADI's construction improperly ignores the patents' description of these issues and the specific approach that the

patents adopt to address them. *See Voice Techs.*, 164 F.3d at 615 (finding the inventor's "explanation of the problems that existed at the time the invention was made and the inventor's solution to these problems" relevant to the claim construction analysis).

To support its construction, ADI relies on the most general definition of "repository" in a non-technical dictionary. *See* Ex. A, Webster's Third New International Dictionary, Unabridged (2002). This general definition improperly divorces the claimed repository from its function and from the purpose of the invention, as disclosed and claimed in the patents-in-suit. *See Phillips*, 415 F.3d at 1321 ("[H]eavy reliance on the dictionary divorced from the intrinsic evidence risks transforming the meaning of the claim term to the artisan into the meaning of the term in the abstract, out of its particular context, which is the specification."); *id.* at 1322 ("[A] general-usage dictionary cannot overcome art-specific evidence of the meaning of a claim term.") (internal quotation marks omitted). Such a broad construction is inconsistent with the plain language of the claims, which use different terms for "relational database" and the "repository."

ADI's construction also conflicts with what the specifications describe as "the present invention," shown in Figure 2. This is improper. *See Verizon*, 503 F.3d at 1308 (holding that when a patent "describes the features of the 'present invention' as a whole, this description limits the scope of the invention"). As discussed above, Figure 2 requires extracting the schema information from the relational database and storing the corresponding metadata in a distinct configuration repository in order to capture the "numerous advantages" of this architecture. *Id.* at 8:1-20, Fig. 2. Indeed, the only embodiment in the patents illustrates the process of creating and saving metadata in a distinct "configuration repository" file. *Id.* at Fig. 58 (saving a "New Configuration Repository"). Accordingly, Oracle and Salesforce's proposed construction of "repository" should be adopted.

B. "compilation of code" ('094 patent, independent claims 1, 8, and 15)

Oracle and Salesforce's Proposed Construction	ADI's Proposed Construction
"translation of all or part of a program expressed in high-level language into a computer program expressed in an intermediate language, an assembly language or a machine language"	"the process where a programmer uses a program that translates source code, a language in which programmers can read and write, into machine code, which is the language a computer can execute"

Claims 1, 5, and 8 of the '094 patent require the step of "automatically . . . developing" a graphical user interface "without compilation of code before runtime." The parties dispute whether the claim term "automatically . . . developing . . . without compilation of code," means, as the plain language provides, that all types of compilation should be avoided (Oracle and Salesforce's construction), or whether these claims broadly cover "automatically . . . developing" that permits certain types of code compilation (ADI's construction). Specifically, ADI's construction would ***permit*** all types of compilation, except compilation to "machine code, which is the language a computer can execute." In contrast, Oracle and Salesforce's proposal would not permit any kind of code compilation, including very common types of compilation such as compiling to assembly language or an intermediate language. Oracle and Salesforce's construction is consistent with the purpose of the '094 patent, which is to avoid the complexity and maintenance associated with code compilation in general. And because the '094 patent makes no distinction between different types of compilation in the claims or specification, Oracle and Salesforce's construction reflects the only correct understanding of the claim term.

Techniques for compiling code were well-known in the field of computer science long before the patents-in-suit were filed. Software engineers typically write code in a high-level language called source code, which is designed to be read and written by humans. Such high-level languages use descriptive variable names (such as "customer_name") and other syntactic

conveniences that make reading and maintaining the code easier. The source code is then compiled (by a software program known as a "compiler") from the source code into a lower-level language. The code may be compiled into "native" machine code, such that it can be executed directly by a particular piece of hardware (e.g., an Intel microprocessor), or it may be compiled into an intermediate language (e.g., for execution on the Java Virtual Machine, a piece of software that emulates a microprocessor). *See* Ex. B, Microsoft Computer Dictionary 252 (4th ed. 1999) ("**Java** *n.* . . . Java programs are ***compiled*** into bytecode, which is not refined to the point of relying on platform-specific instructions and runs on a computer in a special software environment known as a virtual machine.") (second emphasis added). This latter "virtual machine" approach is useful because the source code does not need to be compiled for each machine upon which it runs—it only needs to be compiled once for a virtual machine and can then execute on that virtual machine, which may in turn run on many different types of hardware. *Id.* Although compilation to virtual machines was well-established at the time of the filing of the patents, ADI's construction would permit compilation to such intermediate languages as part of the "automatically . . . developing" step, despite the claims' requirement that compilation not occur.

The claim language states, in absolute terms, that the "automatically . . . developing" step must occur "without compilation of code," regardless of what kind of compilation. Claims 1, 8, and 15 all require software that "automatically" develops a graphical user interface from metadata about a database schema. In each case, the claims clarify that this development process occurs "without compilation of code before runtime." Claim 1 is exemplary:

> Computer software for ***automatically generating*** a graphical user interface for a relational database, said software embodied in a computer readable medium having a set of computer instructions encoded thereon and comprising:
>
> utility software extracting schema information from the relational database and

> ***automatically generating*** corresponding schema and graphical user interface metadata stored in a repository; and
>
> user interface software at runtime ***automatically, and without compilation of code before runtime***, ***developing*** from the metadata a graphical user interface appropriate to the relational database.

'094 patent, at claim 1. The claim language focuses on software that performs certain steps "automatically." The first step extracts schema information and then "automatically" generates certain "metadata." The second step requires "automatically . . . developing" a "graphical user interface" from that "metadata." The second step additionally specifies that no compilation happens before runtime, providing additional clarity what the phrase "automatically . . . developing" means. Thus, the plain reading of the claim requires that the "developing" step proceed "without compilation," regardless of type. Nothing in the claim language suggests that "without compilation of code" applies only to compilation to a machine language and not to compilation to an intermediate language, as ADI's construction specifies.

The specifications confirm the purpose of this limitation was to perform the "automatically . . . developing" step without any type of compilation because compilation adds complexity and maintenance costs to the process. '502 patent at 2:25-33, 7:61-67. Nothing in the specification suggests that some types of compilation would be acceptable. On the contrary, the specifications treat compilation as something to be avoided, no matter whether the compilation is directed to machine code, to virtual machine code, or to some other format. For example, one specific problem the patents identify is the overhead and cost associated with compilation as part of the "automatically . . . developing" step. *Id.* The specifications illustrate this problem in the prior art method for developing a graphical user interface from a database. *Id.* at 2:25-33. If the underlying database structure changes (e.g., by adding a new field to a table), the complexity and maintenance costs associated with prior art development methods,

which involve compiling code every time the database changes, are incurred again:

> A key problem arises with the form-based user interface depicted above in that if a new data element (i.e., field) is added to a table in the database, a new data entry control must be added to the table's corresponding form in the user interface. ***This modification requires further programming effort, additional testing, and a recompilation and redeployment of the user interface.*** So not only is the user interface extremely expensive to develop in the first place, it is also very costly to maintain over the lifetime of the database.

Id. at 2:25-33. In the detailed description of the invention, the specification again emphasizes the need to avoid compilation of code:

> As alluded to above, the ***high cost of developing and maintaining data entry user interfaces in the prior art*** is largely attributed to the tight coupling of data entry forms with the structure of the underlying relational database. ***This problem is further aggravated*** by the common practice of hard coding navigation, business rules and Structured Query Language (SQL) into the source code of the user interface, thus ***requiring a recompilation*** and redeployment of the user interface software when modifications to the database structure and/or changes to business logic are made.

Id. at 7:57-67; *see also id.* at 33:40-43 ("Remember, the underlying premise of the present invention is that computer code is extremely expensive to develop and maintain."). ADI's approach suggests that the development and maintenance costs associated with, for example, compilation to an intermediate language, are not substantial and therefore need not be avoided. But nothing in the real world, or in the patents, supports this conclusion. Thus, performing the "developing" step "without compilation of code," regardless of the type of compilation, is consistent with the purpose of the patents-in-suit, which is to automate the process of producing a graphical user interface from a database without costly development and maintenance steps. *Id.* at 1:52-2:12, 2:25-33, 3:5-8.

The extrinsic evidence also supports Oracle and Salesforce's construction. The 1994 edition of the IBM Dictionary of Computing expressly includes compilation to "an intermediate

language, an assembly language, or a machine language" in its definition of "compile":

> **compile** (1) To translate all or part of a program expressed in a high-level language into a computer program expressed in ***an intermediate language, an assembly language, or a machine language***. . . .

Ex. C, IBM Dictionary of Computing (1994) (second emphasis added). The 1987 version of the same dictionary (which ADI cites in support of a different term) is almost identical:

> **compile** 1. To translate a program expressed in a high-level language into a program expressed in ***an intermediate language, assembly language, or a computer language***.

Ex. D, IBM Dictionary of Computing (1987) (second emphasis added).

Not even the dictionaries upon which ADI relies support its attempt to limit "compiling" to producing machine code ready for execution on hardware. For example, ADI cites the Microsoft Computer Dictionary, which states that "compiling is sometimes used to describe translating any high-level symbolic description into a lower-level symbolic or machine-readable format." Ex. B, Microsoft Computer Dictionary 100 (4th ed. 1999). Even the narrowest dictionary definitions provided by ADI are broader than its proposed construction, because they make clear that compilation covers not only the conversion of source code into machine code, but also into its "equivalents" or "a variation." *Id.* (definition of "compile"); Ex. E, IEEE Standard Glossary of Software Engineering Terminology (1990) (definition of "compiler"). Thus, ADI's own dictionaries do not support the narrow construction of "compile" that it has proposed. Even if they did, ADI has selected a construction of "compilation" that conflicts with the consistent disclosures in the specification that require performing the "automatically . . . developing" step "without compilation of code" of any kind. This is improper. *Phillips*, 415 F. at 1324 (holding that extrinsic sources may be used "as long as those sources are not used to contradict claim meaning that is unambiguous in light of the intrinsic evidence").

Nothing in the claims or the specifications differentiate between one type of compilation and another. This is because it would make little sense to claim an "automatic" step of developing the graphical user interface "without compilation of code," where that phrase applies to one kind of compilation step but not others. All types of compilation contribute to the maintenance and cost problems identified in the specifications. Indeed, nothing in the specification suggests that compilation to an intermediate language would be any less troublesome than compilation to native machine code.

C. "entity" ('502 patent, dependent claims 2, 8, and 14; '094 patent, independent claims 2, 9, and 16)

Salesforce's Proposed Construction[6]	ADI's Proposed Construction
"a representation of a physical database table that is stored in the repository"	"an entity represents anything about which information can be stored in a database, for example, a person, concept, physical object, or event"

Several of the dependent claims of the '502 and '094 patents require "entities," and more particularly, that the claimed "schema and user interface metadata" comprise such entities. The parties dispute whether the "entities" are representations of the physical database tables of the relational database, as the patentee defined in the specification (Salesforce's construction), or whether the entities can represent anything, about which any kind of information, regardless of structure or type, can be stored in a database (ADI's construction).[7]

As discussed above in Section IV, the patents-in-suit are directed to generating metadata that represent the structure ("schema") of a relational database, and storing the metadata in a repository. This is reflected in claim 1 of the '502 patent, which is representative of other

[6] Oracle takes no position as to whether this term should be construed or, if construed, what the proper construction should be.

[7] ADI's proposed construction leaves unclear whether "entities" represent the stored information itself (i.e., "for example, a person, concept, physical object, or event") or instead represent information about the stored information (i.e., metadata).

independent claims in the patents:

> utility software extracting ***schema*** information from the relational database and automatically generating ***corresponding schema and graphical user interface metadata*** stored in a repository

'502 patent, claim 1. Thus, as set forth in the claim language, the corresponding schema and graphical user interface metadata is generated from the extracted schema.

Dependent Claim 2 of the '502 patent then specifies that "entities" are one specific type of metadata representing the schema:

> The computer software of claim 1 wherein ***schema and user interface metadata*** comprise ***entities***, entity fields, entity relationships, and entity search paths.

Id. at Claim 2. The intrinsic record makes clear that "entities" are not simply a representation of any kind of information that could be stored in a database, but instead are a particular type of metadata generated by the claimed software that represents physical database tables.

First, the patentee in this instance acted as his own lexicographer, and provided an express definition of the term "entity" as indicated by the use of quotation marks below:

> The Configuration Repository
>
> Instead of tightly coupling the user interface with the relational database, a configuration repository is used to present a layer of abstraction between the database and the user interface. The configuration repository consists of highly structured metadata. The metadata is organized in the following manner:
>
> Entities
>
> Physical database tables are represented by "entities". A single table may be represented by more than one entity through a technique called "aliasing". For example, separate entities might be defined to distinguish between domestic and international Customers even though a single table, CUSTOMER, is used to store all Customer data.

Id. at 9:45-59; *see Sinorgchem Co., Shandong v. Int'l Trade Co*, 511 F.3d 1132, 1136–39 (Fed. Cir. 2007) ("The term 'controlled amount' is set off by quotation marks—often a strong indication that what follows is a definition."); *Netscape Comm'n Corp. v. ValueClick, Inc.*, 684 F.Supp.2d 678, 688 (E.D. Va. 2009) ("Significantly, the patentee's intention to act as

lexicographer is evidenced by two recognized indicators: (i) the use of quotation marks around the claim term …."). In addition to this express definition, the specification elsewhere confirms, when describing how metadata is generated, ('502 Patent, "Metadata Generation", starting at 17:26), that entities are generated to represent tables: "One entity is ***created*** for ***every table*** in the database." *Id.* at 17:52.

Salesforce's proposed construction is further in accordance with the specification's disclosure regarding "schema," "entities," and related metadata structures stored in the repository. More particularly, "schema" is defined as "collection of database objects that includes tables, views, indexes, and foreign keys." *Id.* at Table 1. "Entities" cannot simply represent any kind of information that could be stored in a database; rather, they must be representative of an element that specifically defines the structure of the database in accordance with the definition of "schema," i.e., database tables.

Next, as referenced above, the configuration repository described above "consists of ***highly structured*** metadata," ('502 patent, at 9:48-49 (emphasis added)), which highly structured metadata include "entities," "entity fields," and "entity relationships" *See generally*, *id.* at 9:51-13:2. Similarly to entities, the specification defines a specific relationship between physical database tables and "entity fields" and "entity relationships," i.e., "'Entity fields represent fields within ***database tables***," *id*. at 10:30, and "Entity Relationships" are "Relationships between ***database tables***," *id*. at 10:46. Thus, each of these other "highly structured" types of metadata—"entity fields" and "entity relationships"—are also defined in terms of "database tables," further confirming that "entities" are representations of "database tables."

ADI's proposed construction ignores the intrinsic evidence. First, there is simply no disclosure in the specification that an entity represents "anything about which information can be

stored in a database," as ADI claims. Second, under ADI's proposed construction, "entities" might not be a form of metadata at all. However, ADI's disclosure of "Intrinsic Evidence" notably omits reference to column 9, lines 45-52—critical context highlighted above confirming that "entities" are a form of highly structured metadata. ADI further omits reference to the statement in the specification that "[o]ne entity is created for every table in the database," *id*. at 17:52, as well as to the related definitions of "Entity Fields" and "Entity Relationship," *id*. at 10:30, 10:46, which likewise make clear an entity is a form of metadata.

Even the truncated intrinsic evidence actually disclosed by ADI—seven lines and two figures—is consistent with or expressly supports Salesforce's proposal that "entities" are a form of metadata that represent database tables. First, ADI itself points to part of the definition of "Entities" that expressly states that "[p]hysical database tables" —and not "anything about which information can be stored in a database" —are "represented by 'entities.'" *Id*. at 9:53-59. Second, ADI points to Figure 4, which depicts an "Entity Relationship Diagram" for an "example database." This figure, however, neither defines "Entity" nor depicts ADI's claimed invention—the applicants neither claim to have invented the "example database" nor this "Entity Relationship Diagram." Rather, the sole named inventor conceded during prosecution of the '502 patent that the "Entity relationship diagrams" that he prepared and filed with the Patent Office depict "portions of the ***metadata*** database" that are "discussed in the ***Configuration Repository*** section of the patent application" – thereby confirming that the entities are metadata. Second Decl. John Kesler at ADI000819 (June, 11, 2004). Finally, ADI points to Figure 61 and "associated text," which undermine ADI's proposed construction. In particular, this "associated text" states: "In Fig. 61, the Refresh process has completed and the Customer Order ***entity metadata*** is shown." *Id*. at 38:64-65. That is, Figure 61 depicts a list of metadata stored in the

repository that corresponds to physical database tables, such as the "Customer Order" table—in accordance with Salesforce's, not ADI's construction.

D. "appropriate to the relational database" ('502 patent, independent claims 1, 7, and 13; '094 patent, independent claims 1, 8, and 15)

Oracle and Salesforce's Proposed Construction	ADI's Proposed Construction
Indefinite	This term is not indefinite. The plain and ordinary meaning governs the construction of this phrase, and it does not need to be construed. If the Court determines construction of this phrase is necessary, ADI proposes that "a user interface appropriate to the relational database" means "a user interface that allows viewing and editing of data in the database"

The independent claims of the patents-in-suit require "developing from the metadata a user interface appropriate to the relational database." Oracle and Salesforce contend that the phrase "appropriate to the relational database" is indefinite because, among other things, it fails to "delineate the scope of the invention using language that adequately notifies the public of the patentee's right to exclude." *Datamize, LLC v. Plumtree Software, Inc.*, 417 F.3d 1342, 1347 (Fed. Cir. 2005) (affirming lower court's grant of summary judgment that sole independent claim is invalid as indefinite due to the phrase "aesthetically pleasing" because the "scope of claim language cannot depend solely on the unrestrained, subjective opinion of a particular individual purportedly practicing the invention"); *Shire LLC v. Teva Pharm. USA Inc.*, C.A. No. 10-329, 2012 WL 975694, *3 (D. Del. 2012) (rejecting plaintiffs' proposed construction of "desired level and duration of behavioral inhibition" under *Datamize* because the construction "does not provide any objective definition identifying a standard for determining when the degree and length of time of behavioral improvement has been reached"). ADI contends the phrase is not

indefinite and does not require construction. In light of the parties' positions and the Court's practice of resolving indefiniteness through summary judgment or at trial rather than through claim construction, Oracle and Salesforce do not request that the Court address the term "appropriate" at this time.

E. Structure For "repository means for containing said metadata" ('502 patent independent claim 7; '094 patent, independent claim 8)

The parties' sole dispute relating to this limitation is whether the corresponding structures for performing the function "containing said metadata" include structures[8] related to filters used for advanced searching against entities stored in the repository. Specifically, a user conducting an advanced search can access a list of filters for searching "entities defined in the search path of the parent entity." '502 patent, 16:19-20. ADI asserts that the portion of the specification stating that such "filters" used for "advanced searching" "may be saved for later reuse using Extensible Markup Language (XML) as illustrated in FIG. 29" is structure corresponding to the claimed function of "containing said metadata." *Id.*at 16:29-31. However, the specification provides no connection between the filters used for searching entities (or the storage of any such filters) and the "containing" of any "metadata" in the repository. As a result, there is no disclosure that these "structures" identified by ADI correspond to the claimed function. *B. Braun Med., Inc. v. Abbott Labs.*, 124 F.3d 1419, 1424 (Fed. Cir. 1997) ("[S]tructure disclosed in the specification is 'corresponding' structure only if the specification or prosecution history clearly links or associates that structure to the function recited in the claim.").

VI. CONCLUSION

For the reasons above, Oracle and Salesforce respectfully request that their proposed claim constructions be adopted by the Court.

[8] The disputed structures are '502 Patent, Fig. 29 (and associated text), Col. 4, Table 1 (entry for "Extensible Markup Language (XML)"), and 6:18-19

MORRIS, NICHOLS, ARSHT & TUNNELL LLP

/s/ Stephen J. Kraftschik

Mary B. Graham (#2256)
Stephen J. Kraftschik (#5623)
1201 N. Market Street
P.O. Box 1347
Wilmington, DE 19899-1347
(302) 658-9200
mgraham@mnat.com
skraftschik@mnat.com

OF COUNSEL:

Jared Bobrow
Sonal N. Mehta
Andrew L. Perito
Justin M. Lee
WEIL, GOTSHAL & MANGES LLP
201 Redwood Shores Parkway
Redwood Shores, CA 94065
(650) 802-3000

Attorneys for Oracle Corporation and Oracle America, Inc.

MORRIS, NICHOLS, ARSHT & TUNNELL LLP

/s/ Jennifer Ying

Jack B. Blumenfeld (#1014)
Jennifer Ying (#5550)
1201 North Market Street
P.O. Box 1347
Wilmington, DE 19899
(302) 658-9200
jblumenfeld@mnat.com
jying@mnat.com

OF COUNSEL:

Kevin P. B. Johnson
Ray Zado
QUINN EMANUEL URQUHART
& SULLIVAN, LLP
555 Twin Dolphin Drive, 5th Floor
Redwood Shores, CA 94065
(650) 801-5000

Edward J. DeFranco
Joseph Milowic III
QUINN EMANUEL URQUHART
&SULLIVAN, LLP
51 Madison Avenue, 22nd Floor
New York, NY 10010
(212) 849-7000

Attorneys for salesforce.com, inc.

Dated: April 8, 2014